XML-Route: A Fault-tolerant Multicast Architecture
by
Qian Z. Wang

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

December 15, 2000

Copyright 2000 Qian Z. Wang. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

David K. Gifford
Thesis Supervisor

Accepted by

Chairman, Department Committee on Graduate Theses

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUL 31 2002
LIBRARIES
XML-Route: A Fault-tolerant Multicast Architecture

by

Qian Z. Wang

Submitted to the
Department of Electrical Engineering and Computer Science

December 15, 2000

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

XML-Route is a fault-tolerant architecture for multicasting time-critical data. It uses a mesh topology coupled with content-based routing and compression to efficiently deliver an uninterrupted stream of XML packets to end clients. XML-Route includes a generalized mesh-building algorithm for dynamically constructing a fault-tolerant multicast fabric. The degree of connectivity in the mesh can be adjusted to provide various levels of fault-tolerance. Associated recovery and renewal algorithms can restore the mesh to a fault-tolerant state after the occurrence of failures. The cost of this fault-tolerance is an increase in bandwidth usage of the multicast network. To minimize the bandwidth premium, we have incorporated XML compression and content-based routing into the XML-Route architecture.

We present the XML-Route architecture and an implementation of XML-Route used to serve a real-time air traffic data stream to end clients. Experimental data show that compressed XML formatted data is only 50-80% of the size of compressed data in their original format. In addition, in typical usage scenarios, content-based routing brings the bandwidth usage of XML-Route to less than 150% of the bandwidth used in a tree topology.

Thesis Supervisor: David K. Gifford
Title: Professor of Electrical Engineering and Computer Science
Acknowledgments

I would first like to thank my family. I want give my deepest thanks to my mother, Xin, and step-father, Fred, who gave me the love and support I needed to complete this work. Without their help I could never have completed this thesis. I also thank my father, Qin, who has given me much love and advice even though we have been half a world apart. I would also like to thank the rest of my family for their unwavering love and faith in me.

I give my gratitude to Professor David Gifford, who has been a mentor and a friend to me these past three years. Without his help and guidance, this thesis would certainly be in a much poorer state. I would also like to thank everyone who has helped me on this project: Jeanne Darling, David Zhang, Micah Gutman, and Cheewee Ang. Thank you for your help and more importantly for your friendship.

Finally I would like to thank all of my friends at MIT and elsewhere. You are the people that make all the difference in life.
Contents

1 Introduction .................................................. 6

2 Related Work ................................................. 8
   2.1 Reliable Multicast Systems ............................. 9
   2.2 Content Distribution Systems ............................ 10

3 XML-Route Architecture ...................................... 11
   3.1 Target Usage and Requirements ......................... 11
   3.2 Application-Level Multicasting ........................... 12
   3.3 Dynamic Mesh Topology ................................... 12
   3.4 XML Streams ............................................. 13
   3.5 XML Content-Based Routing ............................. 14

4 Multicast Mesh Protocols .................................... 15
   4.1 Initialization ............................................. 15
   4.2 Mesh Building Protocol .................................. 16
   4.3 Recovery Protocol ....................................... 18
   4.4 Renewal Protocol ....................................... 21

5 Delivery Protocols .......................................... 22
   5.1 XML Routing Protocol .................................... 22
   5.2 Admission Control Protocol .............................. 23

6 Evaluation .................................................... 24
   6.1 Implementation .......................................... 25
   6.2 Fault-tolerance ........................................... 25
6.3 XML Streams ............................................. 25
6.4 XML Compression ....................................... 28
6.5 XML Routing ............................................. 28

7 Conclusions ............................................. 29
1 Introduction

XML-Route is motivated by the need for a highly-reliable, fault-tolerant architecture to deliver time-sensitive data from the data source to a large number of end clients. In most traditional multicast systems, reliability has meant the ability to detect and recover from node failures. Recovery typically takes the form of retransmission of the lost data. This form of reliability is entirely adequate for most types of data that are delivered by traditional multicast systems. However, for time-critical data-streams, the interruptions and delays introduced in traditional error recovery may be unacceptable. Time-critical data differs from traditional multicast data in that their value depends heavily on their timely delivery to clients. Examples of such data include stock price quotes, air traffic updates, battle field intelligence [11] and data needed for real-time decision making. XML-Route was designed for the delivery of these kinds of data.

We will now briefly define some of the terms we will be using in describing XML-Route.

- **Mesh** A mesh is a directed graph such that at least two nodes in the graph have more than one path between them. We define a mesh with connectivity \( n \) to be a mesh in which every router has \( n \) parents.

- **Router** A router is the basic building block of XML-Route. A router forwards data in a mesh.

- **Root** A root is a router that has a direct connection to the data source. This connection is not shared with any other node, and the roots fail independently.

- **Interior Router** An interior router is a regular router that resides in the XML-Route network.
• **Client** A client is a data sink. It receives data but does not transmit data to other nodes.

One way to insure uninterrupted multicast data delivery in the face of failures is to replicate some or all of the links in a multicast network. However, redundancy generally comes at a cost. By introducing redundant connections, we will use more bandwidth within the underlying network fabric. The presence of failures in the network also means that any static topology will likely degrade over time, leading to a loss of the fault-tolerant properties of the multicast network.

XML-Route marks a departure from other multicast systems by using multicast meshes instead of multicast trees as its delivery fabric. The mesh allows a node in the multicast network to receive data from more than one source, thus preventing the failure of any single node from partitioning the multicast network. Such multicast meshes are automatically constructed and maintained by XML-Route. This automatic and dynamic management of the multicast fabric means that XML-Route has the ability to not only tolerate failures but to recover and restore itself into a fault-tolerant state after failures take place. The self configuration feature of XML-Route also means that failed nodes in the network can simply be replaced by starting up a new node and allowing it to join the network. The failed node can then be repaired off-line without impacting the multicast network. Finally, XML-Route employs novel features such as XML compression and content-based routing to reduce the bandwidth usage of the multicast mesh.

This thesis makes the following contributions:

• A novel multicast mesh topology for high levels of fault-tolerance. We prove properties
of the multicast mesh that give it these fault-tolerant characteristics.

- A generalized algorithm for automatically creating such multicast meshes to achieve an arbitrary level of fault-tolerance.

- Related algorithms for restoring the multicast mesh after failures and for adapting the mesh to changing conditions in the underlying network.

- A novel use of XML streaming and content-based routing to achieve bandwidth savings while adding value to the data stream.

- Results from our implementation that show XML Route be fault-tolerant and efficient.

Section 2 puts XML-Route into the context of prior work. Section 3 discusses the requirements and design choices of the XML-Route architecture. We present the detailed workings of the mesh building and maintenance protocols, along with proofs of the fault-tolerant properties of the mesh in section 4. In section 5 we discuss the details of the streaming and routing protocols. Finally, section 6 presents data from tests on our implementation of XML-Route and result from simulations, which show that XML-Route achieves its goals of fault-tolerance and efficiency.

2 Related Work

XML-Route builds on research in reliable multicast and application-level data distribution systems. The goal of XML-Route is to find a combination of high reliability and efficient distribution. We will discuss work in each area that has influenced the design of XML-Route.
2.1 Reliable Multicast Systems

There is a large body of work on reliable multicast. For example, SRM [5], RMTP [10], RMTP-II [18], TMTP [19], and Totem [12] are just a few of the reliable multicast protocols developed to date. The reliability in these protocols come from a combination of fault detection and error recovery. One class of reliable multicast protocols depend upon the servers to track the error status of clients [10, 19, 7]. These systems rely upon acknowledgment (ACK) packets to signal the successful transmission of data packets. Errors are detected by the senders of the data when they fail to receive ACKs. Error recovery is performed by resending the lost data. Many problems exist for this approach including the inability to scale beyond a handful of nodes. Although systems such as RMTP [10] were built to try to solve some of these problems, the more popular approach to reliable multicast is to make receivers responsible for error detection and recovery. By assigning sequence numbers to data packets or by sending periodic heartbeat messages, receivers can detect errors when gaps are found within the data stream. Negative acknowledgment (NACK) packets are then sent by the receivers to the senders requesting retransmission of the lost data. SRM [5] is one of the best known receiver based protocols for reliable multicast. There are also protocols that require router based support [14, 16, 8], many of which are base on IP multicast [2]. Compared to these reliable multicast protocols, XML-Route has a number of advantages. The vast majority of these protocols require network operator cooperation. Most of these systems handle network packets directly and some of these, such as LMS [14] and PGMG [16], require additional software to be install into network routers. XML-Route, on the other hand, is an application-level system that can be directly deployed over a wide range of networks. More importantly, XML-Route is able to completely mask certain failures. This property gives XML-Route a higher level of reliability than the detection/recovery
scheme common to all of these protocols. Furthermore, XML-Route is a self-configuring system with the ability to adapt to changing network conditions while most current reliable multicast systems have fixed hierarchical structures.

2.2 Content Distribution Systems

A number of content distribution systems have been built to deliver data to a large number of clients [17, 3, 4, 6, 13]. Overcast [6], like XML-Route, is an application-level multicast system that is capable of self-organization and dynamic reconfiguration. Overcast includes a bandwidth optimizing heuristic in its multicast tree building protocol. The heuristic uses observed bandwidth measurements to avoid bandwidth constrained links in the underlying network. XML-Route also makes use of this heuristic in its mesh building protocol. However, Overcast’s reliability features are limited to the replication of the root node and a playback cache at the internal nodes.

The Information Bus (TIB) [13] is a large scale information distribution system that has been used for applications such as delivering stock quotes. The most interesting feature of TIB is the so-called “subject-based addressing” of data. In subject-based addressing, the content producer is called a publisher and the client is called a subscriber. The subscriber employs filters that can distinguish data by subject to collect only the data that it needs. Subject-based addressing is somewhat similar to XML-Route’s content-based routing. However, content-based routing is more flexible in that it is able to inspect any part of an XML-packet and perform complex matching on the data. In contrast, subject-based addressing is limited to distinguishing data by a predetermined subject. Another difference between XML-Route and TIB is that the content-matching in XML-Route is done
by the multicast network while in TIB clients are responsible for filtering their own data. Besides saving bandwidth, XML-Route's router-side matching can serve a wider variety of clients, including low-power and low-bandwidth devices that are incapable of doing their own filtering.

3 XML-Route Architecture

In this section we discuss the design of the XML-Route architecture. First we present the targeted use of the system and the resulting system requirements. We then examine the architectural features of XML-Route that are aimed at satisfying these requirements. We discuss the benefits and drawbacks of each feature.

3.1 Target Usage and Requirements

The most important design requirement of XML-Route is uninterrupted availability. We construe this requirement to mean n-resilience — the ability to withstand n simultaneous node failures without momentary service interruption. Note that link failures are more difficult to handle and require knowledge of the topology of the underlying network. As an application-level system, XML-Route will assume that the underlying network is a best-effort delivery system, which means that in the absence of a network partition, packets will still be delivered.

The solution to the n-resilience problem should be general in the sense that the user should be able to obtain higher levels of reliability. On the other hand, we should not ignore efficiency considerations. In almost all networks, bandwidth is a limited resource. Therefore, the design should attempt to be as efficient in its bandwidth usage as possible without compromising its reliability. Finally, the design of the system should be scalable so that
sufficient resources can be brought to bear on large problems and reasonable efficiency can be attained in small problems.

### 3.2 Application-Level Multicasting

To make XML-Route as widely applicable as possible, we have designed it to be an application-level multicast system for wide-areas networks. The XML-Route architecture can be adapted to work over a variety of underlying networks, hence extending the utility of the system. An application-level system also allows us to incrementally deploy a large scale network and to quickly and easily replace failed units with new nodes.

The primary drawback of an application-level multicasting system as compared to lower level protocols is efficiency. The associated overhead of a higher level system must give up some efficiency in exchange for flexibility. For XML-Route, we believe that the flexibility of an application-level system outweighs the efficiency loss. We also attempt to win back some efficiency elsewhere in the system with innovative design features.

### 3.3 Dynamic Mesh Topology

The independent routing nodes within XML-Route are capable of self-organizing into a multicast mesh. This mesh is dynamic in two senses. First, the topology of the mesh is dependent on not only the underlying network but also on network load at the time the mesh is constructed. Therefore, given identical nodes on an identical network, two iterations of the self-organization phase of XML-Route may produce different meshes. The second sense in which the mesh is dynamic is that XML Route is capable of periodically refreshing the multicast fabric by asking some subset of its nodes to try to reconnect to the network and
possibly find new connection points. This feature allows XML-Route to adjust to changes in network conditions. During the refresh phase, all nodes will continue to receive data.

The main advantage that this mesh topology has over the traditional multicast tree topology is fault-tolerance. By allowing each routing node within XML-Route to connect to more than one parent node, we can eliminate single points of failure from the multicast network. Therefore, if a single node or link failure should occur within the multicast fabric, the mesh topology would be able to mask it while a tree topology would not.

The primary disadvantage of the mesh topology is that it uses more bandwidth within the underlying network than a tree architecture. In a system where every router has two parents, the effective bandwidth usage is twice as much as a tree consisting of the same nodes. The cost goes up linearly with the degree of connectivity of each node. However, by using content-based routing and link compression, we have managed to significantly reduce the bandwidth consumption of XML-Route in most real world scenarios.

3.4 XML Streams

XML-Route uses a stream of “XML packets” for data delivery. An XML packet is a complete XML document containing a logical unit of data. XML is a good streaming format for several reasons. First, XML gives needed structure to streaming data. Typically, streaming formats tend to be either unstructured (such as byte streams) or overstructured (such as MPEG), making it difficult to represent logical units of structured data. XML can preserve the logical structure of the data being streamed, making it easier for the end clients to make sense of the data. Secondly, XML is human-readable and self-documenting. By taking care
to give XML tags descriptive names and defining a clear DTD, we can make the meaning of the data self-evident. In addition, XML is an emerging standard in networking with a wide variety of support tools already available. It is likely that in the near future most platforms will include some default XML parsing capability. By formatting data as XML, we can ensure that clients for XML-Route will be easy to develop. XML also has built-in validation mechanisms, which can be used to ensure XML data packets are well-formed and structurally valid.

The primary disadvantage of XML as a streaming format is that it usually causes an increase in data size. Indeed, in our application uncompressed XML typically causes an increase in representation size when compared to more compact formats (Section 6). However, in our experience XML has proved to be an extremely efficient format for compression, achieving smaller compressed data size than proprietary formats.

3.5 XML Content-Based Routing

The content-based routing system in XML-Route is built upon a predicate language. When a child, which may be an end client or an intermediate node, connects to an XML router, it registers a predicate describing the XML packets it wishes to receive. Each routing node sends each child only the packets that the child desires. Packets that are not wanted by any child are discarded. By not sending unneeded data, the routing node effectively reduces the bandwidth requirements of every node downstream.

Bandwidth reduction is of course not the only benefit of content-based routing. By giving end clients only the data they require, the burden on the clients is reduced since they receive
much less data than they would without content-based routing. This fact is particularly important since a multicast system without storage can only deliver data at the rate of the slowest node. In addition, by relieving clients of the need to filter a large amount of data to get at the data they need, XML-Route can serve a wider range of clients, including those that do not have sufficient power to do their own filtering. Another feature that we have incorporated into XML-Route’s routing system is the ability to limit the visibility of certain data to certain clients. By conjoining a policy predicate with the predicate supplied by a client at connection time, we can protect sensitive data from unprivileged clients.

4 Multicast Mesh Protocols

This section describes how routers within XML-Route automatically build and maintain a fault-tolerant multicast fabric. We describe in detail the initialization, mesh-building, recovery, and renewal protocols in XML-Route that work together to provide the fault-tolerant characteristics of the system. Clients of XML-Route are expected to participate in these protocols to receive the fault-tolerance benefits, although they are not required to do so. Clients are further distinguished from internal nodes so that they are treated as leaf nodes and not reported as potential points of connection in these protocols.

4.1 Initialization

XML-Route supports an arbitrary number of root nodes. These root nodes obtain data through independent channels that do not share a single point of failure. Each internal node in XML-Route must be able to obtain a list of known root nodes upon initialization. The mechanism for obtaining the list of roots include a URL, DNS lookup, or a configuration file.
4.2 Mesh Building Protocol

The number of connections that each node should make when joining the network is specified by a parameter $n$. The number $n$ does not have to be the same for every node. For example, we may have a highly connected core of server nodes with $n = 4$ while clients that connect to these nodes only use $n = 2$ for basic fault-tolerance. The only restriction is that $n$ must be greater than 0 and less than or equal to the number of root nodes in the network, since having $n$ be greater than the number of root nodes will result in a failure to form $n$ distinct connections. The mesh built by this protocol maintains two invariants. The first invariant is that the mesh has a min-cut of $n$, if we consider the root nodes to be a single unit in the mesh. The second invariant is that there are no directed cycles within the mesh. These two invariants imply that a mesh with connectivity $n$ will be able to tolerate $n - 1$ simultaneous node failures without reconfiguration. In other words, an $n$-resilient mesh requires connectivity of $n + 1$. We will prove these invariants at the end of this section. We will also show in later sections that the above invariants are not violated by the mesh maintenance protocols.

The mesh building protocol is described in pseudo-code form in Figure 1. As part of the connection handshake, the node will obtain from each of the parents the $n$ grandparents in that branch, resulting in an $n \times n$ array of grandparents. The node will also receive a depth figure from each of its parents and will set its own depth to 1 more than the maximum parent depth received. The grandparents array and depth will be used in the recovery protocol discussed in the next section.

We will now show that the min-cut of $n$ invariant and the no directed cycles invariant hold for all meshes constructed by the above protocol. The proof for no directed cycles is
Inputs:

roots  /* list of root nodes */
n  /* degree of connectivity */

Variables:

scan-list  /* list of nodes to scan*/
parents  /* list of 3-tuples of the form (node, bandwidth, status) */

Begin mesh building:

scan-list := roots

For i from 0 to n - 1, parents[i] := (node: null, bandwidth: 0, status: “not done”)

WHILE scan-list ≠ empty list DO

   FOREACH scan-node IN scan-list DO

      bw := measure-bandwidth(scan-node)

      test-tuple := (node: scan-node, bandwidth: bw, status: “replacement”)

   FOREACH parent-tuple IN parents DO

      IF bw ≥ parent-tuple.bandwidth

         THEN parents.insert-at(parents.index-of(parent-tuple), test-tuple)

         parents.remove-last()

         EXIT FOREACH

      FI

   OD

FI

OD

FOREACH parent-tuple IN parents DO

   IF parent-tuple.status = “not done”

      THEN parent-tuple.status := “done”

   ELSE IF parent-tuple.status = “replacement”

      THEN scan-list := get-parent-list(parent-tuple.node)

      EXIT FOREACH

   FI

FI

OD

Finish mesh building: connect to every node in parents

Figure 1: The mesh building protocol of XML-Route
obviously true. A directed cycle can only form if a node is connected to one of its own descendants, and no new node has any descendants in the network. It is therefore not possible to form a directed cycle during mesh building.

The proof for the min-cut of $n$ property is by induction. We know that there are at least $n$ root nodes and each root node has an independent path to the data source. The base case of our induction proof is adding one node to the network. This node will construct a “parents” list of length $n$ drawn from the list of root nodes. Since in the course of selecting parent nodes, we never consider any node more than once, each node in the “parents” list is distinct. So the first node in the network will be connected to $n$ distinct root nodes, which clearly results in a min-cut of $n$ for the node that we added.

Now we will assume that when there are $m$ non-root nodes in the multicast network and the min-cut is still $n$. When we add the $m+1$st node we again construct a “parents” list of $n$ nodes. As we still only consider each node that we encounter once, and there are no directed cycles in the network, each node in the “parent” list is distinct. So the new node is connected to $n$ distinct nodes. Since the new node is not connected to one of its own descendants, a cut separating it from the rest of the network has value $n$. From this induction step, we can conclude that the min-cut of the network is still $n$.

4.3 Recovery Protocol

In the previous section we proved that the meshes built by XML-Route have the desired fault-tolerant properties. A mesh with connectivity of $n$ can tolerate and mask the effects of $n - 1$ simultaneous node failures. However, after such a failure, the mesh is left in a vulner-
able state which can no longer tolerate \( n - 1 \) simultaneous failures. XML-Route therefore includes a fault detection and recovery protocol to restore the mesh into a fault-tolerant state after one or more node failures.

The fault detection scheme is based on a node detecting the failure of one of its parents. Once a node detects a fault in one or more of its parents, it will simply look up the \( n \) parents of each of the failed nodes from the grandparents array. It is possible that some of its grandparents are also already its parents via another connection. These nodes will be eliminated from the lists. For each failed parent, the node will then have a list of eligible grandparents. We will show in a moment that there is always at least one eligible grandparent for every failed non-root parent. From each list, the node will pick the grandparent to which it has the highest observed bandwidth, measured in the same way as before, and form a new connection. It will also update the grandparents array and the depth of the current node accordingly. The only exception to this protocol occurs when a root node fails. In this case, there are no parents for the recovery connection. The affected nodes will attempt to perform the renewal protocol discussed in section 4.4 until another appropriate node is found or the failed root node is replaced.

We know that there will be at least one eligible grandparent for every failed non-root parent because in the absence of directed cycles, a parent node and a child node can share at most \( n - 1 \) grandparent nodes. The child may be connected to \( n - 1 \) of its parent’s parents, but there is always one grandparent that is not shared. This fact allows us to bypass the more comprehensive tree building protocol while recovering from faults. The recovery protocol is thus built to minimize the amount of time the multicast network spends in a vulnerable
state. The time it takes to detect a fault and recover from it, $t_r$, determines the types of consecutive faults that XML-Route can handle. XML-Route is able to tolerate groups of $n - 1$ simultaneous faults that that occur at least $t_r$ apart, provided that there are still enough nodes left in the multicast network to satisfy a connectivity of $n$ for every interior node.

It is easy to see that the recovery protocol does not violate the invariants of the mesh. Since the recovery protocol restores the $n$ distinct connections to every node, the min-cut of the mesh will still be $n$. Also, since the new connections are made only to the grandparents of the affected nodes, the recovery protocol cannot introduce directed cycles into the network.

One final concern for the recovery protocol involves the possibility that a parent might become overloaded when one of its children fails and the affect grandchildren all choose it as the recovery node. We first note that these grandchildren will not cause the router to receive more data since the router already receives a superset of the data required by the grandchildren. The concern is for processing resources to transmit data to the additional children, which we can address by asking each node to reserve at least half of its processing capacity for use in recovery. Assuming most internal routing nodes have similar processing power, this conservative policy should guarantee that a parent will be able to handle its grandchildren in the event that one or more of its children fails. In reality, the nodes are more likely to be bandwidth bound in terms of the number of children they can accept, so the additional processing incurred during the recovery should not be a serious issue in most situations.
4.4 Renewal Protocol

Each node within XML-Route will periodically try to reevaluate its connections in order to respond to changing network conditions. The time at which a node will attempt this reevaluation can be chosen at random within a certain range to ensure that the network is not overwhelmed by too many nodes renewing at once. These time intervals can be negotiated at the time a node connects to the XML-Route network or they can be globally predetermined based on the expected number of nodes in the network.

The renewal protocol is essentially the mesh building protocol with a few additional restrictions. In the normal case, the renewal protocol begins by selecting the parent with the lowest bandwidth as a candidate for replacement. The node then performs the mesh building protocol with $n = 1$, thus finding a renewal candidate node. One of the additional restrictions is that while finding the renew candidate, we must check each candidate against the parents list to make sure that it is different from all of the existing parents. When the new node is finally found, we can compare it to the lowest-bandwidth parent and replace it if necessary. Note that in this case, the mesh remains in a fault-tolerant state while the renewal process is taking place. XML-Route will also run the renew protocol after one or more root node failures. The children of the failed root nodes will perform a renewal rather than a recovery. The renew procedure is exactly the same as above with the exception that the failed parents are the ones to be replaced and $n$ will be set to the number of parents that need to be replaced.

In the renew phase, unlike in the initial mesh building, a directed cycle could be formed if the replacement node is one of the current node’s descendants. Since the current node
might have a large number of children and there are always $n$ ways to reach any node, it could be very inefficient to check for cycles. Therefore, to avoid directed cycles, we will make use of the level figures collected during mesh building. Recall that the root nodes have level 0, and each internal node have a level that is $1 + \max(L_1, \ldots, L_n)$ where $L_i$ are the levels of its parents. In the renewal protocol, we place the additional restriction that the replacement node must have a level that is less than or equal to the node's current level. This restriction will ensure that a node will not somehow connect to one of its descendants during renew, thus preventing the formation of any directed cycles.

5 Delivery Protocols

This section describes how data is transported between the nodes within XML-Route and between XML-Route nodes and end clients. The significant features are a compressed XML streaming format, an efficient predicate evaluation system, and an admission control protocol. Together, these features aim to maximize the available resources to serve as many clients as possible.

5.1 XML Routing Protocol

The content-based routing system in XML-Route is based predicates to describe flows. Predicates are defined using a predicate evaluation system that is application independent. For our implementation of XML-Route, a predicate language is defined over the air traffic data being delivered. Each predicate is a disjunctive normal form consisting of terms operating on fields with the XML packets.

When a client connects to a routing node in the multicast network, it registers its predicate
with the routing node. Each non-root routing node in the network also maintains its own predicate which it has registered with its parents. When a new clients joins the routing node, the routing node must expand its own predicate to include any new data needed by the new client. Therefore, the routing node disjoins the terms of the new predicate to its existing predicate and re-registers it with its parents. At this point, the routing node will begin to receive data that the new client needs. Note that in this system, every bit of data that passes through the router is called for by some end client, so no bandwidth is wasted on unwanted data.

Each incoming XML packet is only evaluated once per unique predicate term, resulting in a very efficient routing system. The predicate evaluation system at each routing node takes the form of a tree. The leaf nodes are the individual unique terms from the client predicates. Incoming XML packets are only evaluated on these leaf nodes. The results of predicate evaluation at the leaf nodes are then propagated up through a logic tree where receivers representing clients await the results.

5.2 Admission Control Protocol

XML-Route includes an admission control protocol based on bandwidth estimates. When a client is connected to a node, it sends both the predicate it wishes to register and the observed bandwidth to the node, which was obtained during the mesh-building protocol. The node then performs an estimate on the peak bandwidth required by the client’s predicate. If this peak is below the observed bandwidth reported by the client, then the client is allowed to connect. If the peak turns out to be higher than the observed bandwidth, an "insufficient bandwidth" message is sent to the client.
The estimate of the peak bandwidth requirement for a predicate can be done in many ways. In cases where statistical properties of data being delivered are well known, an estimate can simply be based on a direct analysis of the predicate. For our implementation of XML-Route, we employ a historical database of the flight traffic data to help in the estimation of the peak bandwidth.

Note that this admission control protocol is voluntary since the routing node trusts the client to report its observed bandwidth. However, if a routing node wishes to only accept clients that have sufficient bandwidth to handle the peak data rate, the routing node can take on the responsibility of measuring the bandwidth to the client. Our system implements the voluntary protocol because it provides greater flexibility to the clients. If a client reports its observed bandwidth faithfully and is admitted, then it can be confident that its probability of losing data due to lack of bandwidth is very low. However, if a client is unconcerned about the possibility of data loss due to bandwidth shortage, it can still obtain service by reporting an inflated bandwidth figure. Such “deception” can be useful for clients that would rather receive some data than no data at all.

6 Evaluation

XML-Route has been used to implement a real-time flight information system for all air traffic over North America. Air traffic data is well suited to the XML-Route architecture since it is time-critical and fits well with XML streaming. In this section we briefly describe our implementation and discuss some of our experiences with it. We present test data of the mesh protocols, XML compression, and predicate evaluation system.
6.1 Implementation

Our implementation of XML-Route consists of routing nodes and several different clients all written in Java. We chose Java for its ease of development and cross-platform support. We do sacrifice some measure of performance, but even so our tests show that XML-Route is quite efficient. Our implementation includes all features described in the preceding sections.

6.2 Fault-tolerance

Since we have proved the fault-tolerant properties of the multicast mesh built by XML-Route, we performed empirical tests on our implementation of XML-Route. First, we verified that the mesh building algorithm does work by initializing about a dozen nodes over the Internet. With two roots and \( n = 2 \), the routers were able to organize themselves into a multicast mesh and deliver data. We then introduced artificial node failures by shutting down several of the nodes at 1 minute intervals. We observed the data coming out of the affected nodes and indeed there were no interruptions in the data stream. We also observed that the recovery protocol worked correctly. We performed several variation of this test with different connectivity. We plan to perform more tests on a larger scale in the near future.

6.3 XML Streams

In our implementation each XML packet is an independent status update message on a particular flight. Making each XML packet a complete XML document is not the only choice for XML streaming. It has been suggested that one can make the entire data stream a single XML document where each packet of data is defined as a part of the large document [15]. While this approach is slightly more efficient in that it does not send an XML header
Figure 2: The same flight data formatted in ASDI and XML

for each packet, we decided in favor of independent XML documents for several reasons. In the first place, when compressed, the efficiency advantage of the single document approach is largely negligible. In our data stream, the difference was 6% before compression and only 1% after compression. Additionally, the independent document stream allows each XML packet to be individually validated according to its own DTD, thus allowing the option to mix several different types of data into the same stream. The single document approach, on the other hand, cannot be validated since the client does not receive a complete document until the end of the connection. Furthermore, the single document approach can result in a more complex DTD for the entire stream. Based on these advantages and the equivalent efficiency, we have chosen to make our XML packets independent XML documents.
Figure 3: Uncompressed Air Traffic Data

Figure 4: Compressed Air Traffic Data
6.4 XML Compression

The data arrives at the XML-Route root servers in a proprietary format [1]. We reformat each message into XML according to a DTD that we have defined. Figure 2 shows an example of a flight status message in its original form, in XML form.

Our implementation of XML-Route uses Java’s built-in compression library base on zlib. Zlib uses a variant of the LZ [20] compression algorithm. As earlier work [9] had shown that static XML documents are highly compressible using zlib, we were interested in seeing if that efficiency would carry over to streaming XML. Indeed, we achieved very encouraging compression results using the standard zlib compressor. Figure 3 shows the uncompressed data rates within a 24-hour period from our air traffic data stream. As the graph shows, before compression, XML formatting caused an increase ranging from 200-300% in the data size over the native ASDI format. Figure 4 shows the compressed data rates for the same 24-hour period. As we can see, the compress XML data size is in fact 20-110% smaller than the compressed ASDI data size. We plan to test streaming XML compression on a wider variety of data types to determine its potential more fully.

6.5 XML Routing

The content-based routing capability of XML Route has the potential of providing significant bandwidth savings. In the most extreme case, when every client is requesting the full stream, an XML-Route network with $n = 2$ consumes twice the bandwidth of a tree-based network. However, in most real world applications of XML-Route, we expect clients to request a relatively small portion of the entire data feed. In these cases, XML-Route’s bandwidth usage is significantly less. For example, in a tree multicast system that sends
the full data stream to every node, the bandwidth used is:

\[(\text{number of links in the tree} \times \text{bandwidth used per link})\]

In XML-Route, the bandwidth used is:

\[\sum_i 2 \times (\text{bandwidth used by the } i\text{th node})\]

Clearly, if every node in the XML-Route network only requests half of the contents of the data feed, XML-Route uses no more bandwidth than the tree network. In our experience, actual bandwidth usage in XML-Route nodes is proportional to the distance of that node from the roots. An average of the usage pattern on our system over the past month shows that XML-Route typically consumes 70-145% as much bandwidth as a tree multicast system would serving the same data.

7 Conclusions

We have described protocols to build and maintain a fault-tolerant multicast mesh. These protocol can be used to build meshes of connectivity \(n\) that are able to tolerate \(n - 1\) simultaneous node failures. We have also described a compressed XML streaming protocol and a content routing system that help to reduce the bandwidth used by the mesh to very reasonable levels. Based on tests on our implementation and our simulation results, we have shown XML-Route to be an efficient fault-tolerant architecture for multicast. We believe that XML-Route would make a good distribution system for time-critical data streams.
References


